Partielle Lösungen zur „allgemeinen“ Problematik

Eine grundlegende Einführung

Florian Rittberger

BACHELORARBEIT

eingereicht am
Fachhochschul-Bachelorstudiengang
Software Engineering

an der
Fachhochschule Oberösterreich
in Hagenberg

2025

Betreuer*in: Andreas Johann Scheibenpflug BSc MSc

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt. Die vorliegende, gedruckte Arbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Hagenberg, am 1. Juli 2025

Florian Rittberger

Inhaltsverzeichnis

Vorwort

Kurzfassung

Maximal 1-seitige „Zusammenfassung“ der Arbeit in Deutsch.

Abstract

This should be a 1-page (maximum) “summary” of your work in English.

Kapitel 1

Grundlegende Konzepte und Technologien

1.1 Backtesting

Backtesting bietet die Möglichkeit den Erfolg einer Investitionsstrategie an historischen Finanzdaten zu testen. Die Erfolgsquote dieser Strategie lässt sich somit einschätzen, ohne dafür spekulativ Geld in aktuelle Märkte investieren zu müssen. Dabei wird eine Strategie zu konkreten Zeitpunkten in gewissen Intervallen befragt, ob in diesem Moment gekauft, verkauft oder gehalten werden soll. Die Strategie erhält aber für die Berechnung der Entscheidung nur Informationen, die zu diesem Zeitpunkt auch tatsächlich verfügbar gewesen wären. ~~Also~~ zukünftige Informationen, die zu diesem Zeitpunkt noch nicht bekannt waren, dürfen nicht in die Entscheidungsfindung mit einfließen, obwohl diese verfügbar sind. [4]
Das Ergebnis eines Backtests kann durch verschiedenste Kennzahlen gemessen werden. Beispielsweise dem prozentuellen Gesamtgewinn oder dem Sharpe-Ratio, welcher das Verhältnis von Rendite zu Risiko misst.

$$\textit{Sharpe-Ratio} = \frac{R_p - R_f}{\sigma_p}$$

- R_p = Rendite des Portfolios
- R_f = Risikofreier Zinssatz
- σ_p = Standardabweichung der Überschussrendite des Portfolios

Nur weil eine Strategie erfolgreich im Backtesting war, heißt dies aber nicht, dass diese auch in der Zukunft erfolgreich sein wird. Ein möglicher Grund hierfür könnten Verzerrungen (Biase) in der Strategie oder bei den Testdaten sein, die während des Backtests einbezogen wurden. Das Ergebnis erscheint dadurch im Test überzeugend, in der Praxis aber wird die Strategie den Erwartungen voraussichtlich nicht gerecht. [3]

1.2 Biase in Backtesting

Durch falsches Anwenden eines Backtesters können verschiedene Biase in die Ergebnisse mit einfließen und diese dadurch verfälschen. Nachfolgend werden vier bekannte Biase beschrieben und wie diese vermieden werden können.

1.2.1 Look-Ahead Bias

Investitionstrategien die Informationen verwenden, die diese zu dem Zeitpunkt der Entschleunigung noch nicht haben konnten, verfallen dem Look-Ahead Bias. Dieser verfälscht das Backtestergebnis. Wenn die Strategie die Aktienkurse der folgenden Jahre in die Entscheidungsfindung mit einfließen lässt, kann diese Aktien, bei denen ein Kursanstieg erwartet wird, kaufen und Aktien, bei denen ein Kursrückgang erwartet wird, verkaufen. Somit wird die Strategie im Backtest einen Gewinn erzielen, aber bei realen Bedingungen, bei denen nicht in die Zukunft geblickt werden kann, versagen. Um dies zu verhindern, sollten ausschließlich Informationen berücksichtigt werden, die zum Zeitpunkt der Entscheidung tatsächlich vorlagen. [11]

1.2.2 Overfitting Bias

Der Overfitting Bias entsteht, wenn die erstellte Handlungsstrategie zu stark an die ausgewählten historischen Daten angepasst wurde, um bessere Ergebnisse zu erzielen. Dadurch wird die Strategie nur auf dieses spezielle Datenset gute Ergebnisse liefern, in der Praxis aber wird dadurch voraussichtlich die Erwartung nicht erfüllt. Ein Beispiel für eine angepasste Strategie sind statisch definierte Kauf- und Verkaufsgrenzen. Um dies zu vermeiden, sollte die Strategie auf mehreren verschiedenen Datensets getestet werden. Ebenfalls sollte zur Erstellung der Strategie ein unterschiedlicher Teil der historischen Daten verwendet werden als zum Testen der Strategie. [4]

1.2.3 Survivorship Bias

Der Survivorship Bias entsteht, wenn in den historischen Daten nur Firmen enthalten sind, die während des beobachteten Zeitraums weiterhin bestanden haben. Firmen, die in der Vergangenheit in Konkurs gegangen sind oder aus anderen Gründen nicht mehr existieren, sind in den Daten deswegen nicht enthalten. Somit wird die Strategie im Backtest nur auf Firmen angewandt, die es geschafft haben zu überleben und dabei wird nicht getestet, ob die Strategie auch bei Firmen erfolgreich wäre die in der Vergangenheit gescheitert sind. Um Verzerrungen zu vermeiden, sollten historische Daten einbezogen werden, die auch Unternehmen berücksichtigen, die im betrachteten Zeitraum insolvent geworden sind. [9] [10]

1.2.4 Data-Snooping Bias

Ein Testergebnis fällt in den Data-Snooping Bias, wenn für einen Test eine zu kleine Auswahl an unterschiedlichen Kursentwicklungen in den Testdatensets verwendet wurde. Daraus entsteht das Problem, dass die Strategie nur auf die in den Datensets vorkommenden Kursentwicklungen optimiert wurde, ohne auf andere Kursentwicklungen Rücksicht zu nehmen. Wenn alle verwendeten Kurse beispielsweise nur von Firmen stammen, die alle in dem betrachteten Zeitraum einen wirtschaftlichen Aufschwung hatten, wird die Strategie auch nur in ähnlichen Szenarien gute Ergebnisse liefern. Somit wird die Strategie in anderen Marktsituationen möglicherweise nicht die Erwartungen erfüllen. Um dies zu vermeiden, sollten Datensätze gewählt werden, die unterschiedliche Aufschwünge und Krisen aufzeigen, um verschiedene Marktsituationen abzudecken. [28] [22]

1.3 State of the Art Backtester

~~Zurzeit~~ gibt es bereits zahlreiche Backtester von den verschiedensten Anbietern. Ein Großteil dieser Systeme ist jedoch kostenpflichtig und laufen nicht lokal, sondern nur auf den Servern der einzelnen Anbieter. Somit werden auch selbst entwickelte Strategien an diese Dienstleister weitergegeben. Diese könnten dann die Ideen hinter den Ansätzen kopieren und für ihre eigenen Zwecke verwenden. Dies ist ein Sicherheitsrisiko für nicht veröffentlichte Strategien. Darüber hinaus ist bei solchen Anbietern häufig auch nicht transparent, wie die Backtester intern implementiert sind.
Im Rahmen dieser Arbeit werden daher nur lokale open-source Backtester analysiert. Dadurch lassen sich die genannten Probleme vermeiden und die Backtester im Detail untersuchen und besser miteinander vergleichen. Nachfolgend werden drei bekannte open-source Backtester beschrieben und deren Aufbau und Funktionsweise erläutert.

1.3.1 Backtrader

Backtrader ist ein in Python implementierter Backtester. Alle Komponenten des Backtesters sind in einzelne Klassen aufgeteilt. Eine Klasse für Finanzdaten, Backtesteinstellungen und Startparameter, eine für die Strategie, eine für einen Kauf oder Verkauf und eine Klasse für alle errechneten Ergebnisse. Um selbst Strategien zu implementieren, muss eine Strategieklasse erstellt werden. Die eigene Strategieklasse muss von einer Basisklasse erben und gewisse Funktionalitäten implementieren. Von der Art der Strategie ab hängt, ob alle oder nur ein Teil der Funktionen implementiert werden muss. Mit geerbten Funktionen können beispielsweise Kauf- und Verkaufssignale an den Backtester gesendet werden.
Backtrader bietet keine graphische Benutzeroberfläche, um Backtests zu starten, durchzuführen oder zu analysieren. Der Backtest wird durchgeführt, indem die selbst erstellte Python-Datei ausgeführt wird. Als Datenquelle können CSV-Dateien oder Daten von Online-Quellen wie Yahoo Finance verwendet werden. Alle diese Quellen müssen in einen bestimmen Format vorliegen. Die Resultate des Backtests werden von Backtrader nicht automatisch visualisiert. Stattdessen müssen die Kennzahlen selbstständig aufbereitet und dargestellt werden. Dadurch gestaltet sich die Präsentation aufwendiger, jedoch wird diese dadurch individuell anpassbarer. Zur Veranschaulichung steht eine Hilfsklasse zur Verfügung, die die Darstellung erleichtert und sich zudem durch Ableitung mit eigenen Grafiken erweitern lässt. Ebenfalls vereinfacht die Bibliothek PyFolio die Visualisierung der Ergebnisse. Mit dieser werden aus den Resultaten verschiedene Diagramme und Tabellen mit Kennzahlen erstellt. [37] [26] [33]
Die Performanz von Backtrader ist durch die Geschwindigkeit von Python begrenzt. Backtrader biete keine Möglichkeit die langsamsten Teile der Strategie in einer anderen Form zu implementieren, um die Ausführungsgeschwindigkeit zu beschleunigen.

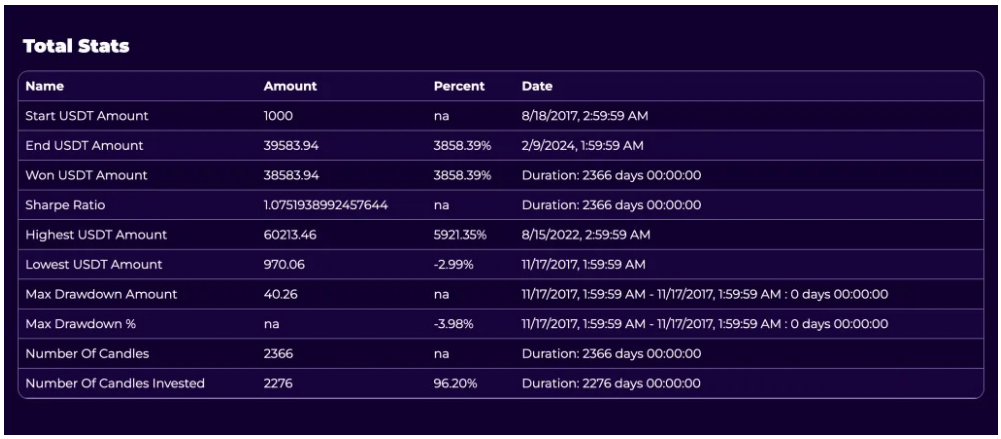
1.3.2 Zipline

Zipline ist ein in Python implementierter Backtester. Dieser wurde ursprünglich von Quantopian entwickelt und wird jetzt von der Community weitergeführt, da 2020 Quantopian den Betrieb eingestellt hat. Die weiterentwickelten Versionen von Zipline haben das Ziel, sich von den ursprünglichen Services von Quantopian zu lösen und die Mög-

lichkeit zu bieten, den Backtester lokal auszuführen. Zwei der bekanntesten Forks sind Zipline Trader und Zipline Reloaded. Den Entwicklern ist es gelungen, sich von den eingestellten Services zu lösen und eigene Funktionalität weiter hinzuzufügen. Beide Forks sind Open-Source und können kostenlos verwendet werden. Ebenfalls ist es möglich beide Systeme über die Kommandozeile zu bedienen, mit Python-Skripten zu steuern oder diese in einem Jupiter Notebook anzuwenden. Zipline Trader geht mehr in die Richtung von Python Skripten, während Zipline Reloaded mehr für die Verwendung in einem Jupiter Notebook oder der Kommandozeile ausgelegt ist.
Damit eigene Strategien verwenden werden können muss eine Funktion implementiert werden, die eine Sammlung an Daten nimmt und Kauf- und Verkaufssignale zurückgibt. Dabei können verschiedenste Hilfsfunktionen und Indikatoren verwendet werden, die in Zipline bereits implementiert sind. Zipline Trader benötigt, um historische Daten zu laden eine Verbindung zu Alpaca oder Interactive Brokers und eine Postgres-Datenbank. Zipline Reloaded hat standartmäßig Kurse für gewisse Aktien vorinstalliert und kann auch Daten von NASDAQ im pickle Format über die API nachladen.
Die Ergebnisse eines Backtests werden in beiden Systemen automatisch generiert aber müssen manuell visualisiert werden. Dafür können verschiedene Bibliotheken wie Matplotlib, PyFolio oder Quantstats verwendet werden. [34] [19] [36] [35] [20]

1.3.3 BacktestJS

BacktestJS ist ein in TypeScript und JavaScript geschriebener Backtester. Dieser kann nur aus der Kommandozeile heraus verwendet werden und hat keine grafische Benutzeroberfläche. Die Ergebnisse können sich aber in einem Webbrowser angesehen werden. In diesem Programm ist es auch möglich eigene Strategien testen zu lassen. Dazu muss eine Strategie in JavaScript oder TypeScript implementiert werden die gewisse Bedingungen erfüllt. Diese Regeln bestimmen wie die Funktion zu heißen hat, welche Parameter diese annimmt, welche Imports nötig sind und wo diese sich im Dateisystem zu befinden hat. Ebenfalls stellt BacktestJS die Möglichkeit zur Verfügung sich historische Finanzdaten in eine lokale Datenbank zu laden und diese als CSV-Datei zu exportieren. Diese Daten stammen von Binance und sind kostenlos verfügbar. Andere CSV-Dateien in diesem Format können auch direkt in die Datenbank mit BacktestJS geladen werden. Damit ist es möglich seine Strategien auf verschiedene Kurse zu testen. [32] [6]
Ein Beispiel für ein Ergebnis eines Backtests mit BacktestJS ist in der Abbildung 1.1 zu sehen. In dieser Statistik werden verschiedene Kennzahlen wie beispielsweise der Gesamtgewinn, der Startbetrag, der Höchststand der Aktie und der Sharpe-Ratio angezeigt. Es werden nach einen Backtest auch noch weitere Tabellen und Diagramme mit genaueren Informationen generiert und angezeigt.
Die Geschwindigkeit von BacktestJS ist limitiert durch die Ausführungsgeschwindigkeit von JavaScript. Es ist nicht möglich rechenintensive Teile der Strategie in einer anderen schnelleren Programmiersprache wie C oder C++ zu schreiben, um die Backtest-Prozess zu beschleunigen. Dies schränkt den Rahmen für Optimierungen der Laufzeit bei komplexeren Strategien ein.



Total Stats

Name	Amount	Percent	Date
Start USDT Amount	1000	na	8/18/2017, 2:59:59 AM
End USDT Amount	39583.94	3858.39%	2/9/2024, 1:59:59 AM
Won USDT Amount	38583.94	3858.39%	Duration: 2366 days 00:00:00
Sharpe Ratio	1.0751938992457644	na	Duration: 2366 days 00:00:00
Highest USDT Amount	60213.46	5921.35%	8/15/2022, 2:59:59 AM
Lowest USDT Amount	970.06	-2.99%	11/17/2017, 1:59:59 AM
Max Drawdown Amount	40.26	na	11/17/2017, 1:59:59 AM - 11/17/2017, 1:59:59 AM : 0 days 00:00:00
Max Drawdown %	na	-3.98%	11/17/2017, 1:59:59 AM - 11/17/2017, 1:59:59 AM : 0 days 00:00:00
Number Of Candles	2366	na	Duration: 2366 days 00:00:00
Number Of Candles Invested	2276	96.20%	Duration: 2276 days 00:00:00

Abbildung 1.1: Beispielergebnis eines Backtests mit BacktestJS [5]

1.4 Bewährte Ansätze für Handelsstrategien

Es gibt viele bereits etablierte Ansätze für Anlagestrategien. In dieser Arbeit werden keine neuen Ansetzte entwickelt, sondern bestehende Ansätze verwendet, diese in JavaScript und C implementiert, in einem Webbrowser ausgeführt und die Ausführungsgeschwindigkeit getestet. Nachfolgend werden drei bereits bekannte Ideen für Investitionsstrategien beschrieben.

1.4.1 Mittelwertrückkehr mit Bollinger Bänder

Die Idee hinter der Mittelwertrückkehr ist, dass der Kurs einer Aktie immer wieder zu einem längerfristigen Mittelwert zurückkehrt. Dabei werden Abweichungen von dem Mittelwert als Kauf- oder Verkaufssignal verstanden. Es werden die Aktien verkauft, wenn der Kurs über dem Mittelwert steigt und gekauft, wenn der Kurs unter dem Mittelwert fällt. [23]
Um die Signale zu verstärken und nicht bei den kleinsten Änderungen eine Aktion auszuführen werden häufig Bollinger Bänder verwendet. Diese bestehen aus drei Komponenten.

- Einem gleitenden Durchschnitt des Kurses über einen bestimmten Zeitraum.
- Einem oberen Band, welches eine bestimmte Anzahl an Standardabweichungen über dem Durchschnitt liegt.
- Einem unteren Band, welches eine bestimmte Anzahl an Standardabweichungen unter dem Durchschnitt liegt.

Somit wird ein breiter Mantel über den Aktienkurs gelegt, der die normalen Schwankungen des Kurses umhüllt. Wenn der Kurs das Band nach oben hin durchbricht, wird dies als Verkaufssignal gewertet. Unterschreiten des Bandes wird als Kaufsignal gewertet. [7] [27] [21]

Wechselkurse von Währungen dienen als Beispiel dafür, dass Kurse oft zu einem Mittelwert zurückkehren. Diese pendeln häufig um einen Mittelwert und bleiben im Verhältnis zueinander stabil. [13]

Bei Strategien mit Mittelwertrückkehr kann es zu Problemen kommen, wenn der Kurs nicht zum Mittelwert zurückkehrt, sondern sich stark in eine Richtung bewegt. Beispiele dafür sind Firmen, die in Konkurs gehen oder einen starken wirtschaftlichen Aufschwung erleben. [25]

1.4.2 Gleitende Durchschnitts Überscheidungen

Gleitende Durchschnitts Überschneidungen Strategien sind eine der bekanntesten und am häufigsten verwendeten Ansätze im Bereich des automatisierten Handels. Dabei werden zwei gleitende Durchschnitte mit unterschiedlichen langen Zeiträumen errechnet und miteinander verglichen. Wenn der kurzfristige Durchschnitt den langfristigen Durchschnitt von unten nach oben durchbricht, wird dies als Kaufsignal gewertet. Wird der langfristige Durchschnitt aber von oben nach unten durchbrochen, wird dies als Verkaufssignal gewertet. Der Unterschied in den verschiedenen Strategien mit dieser Vorgehensweise liegt in der Wahl der Zeiträume für die beiden gleitenden Durchschnitte und der Mehrgewichtung aktueller Werte. [18] [8]

Eine optimierte gleitende Durchschnitts Überschneidungsstrategie erzielt mehr Rendite als die Strategie eine Aktie zu kaufen und zu warten. [14]

Ein Problem bei gleitenden Durchschnitts Überschneidungsstrategien ist, dass es zu Fehlsignalen kommen kann, wenn die beiden Durchschnitte sich häufig überschneiden. Dies passiert häufig in Seitwärtsmärkten, bei denen der Kurs keine klare Richtung hat und sich nur wenig verändert. Dabei kann es dazu kommen, dass es sehr viele Kauf- und Verkaufssignale gibt, die einerseits zu hohen Transaktionskosten führen und andererseits bei langsam fallenden Kursen zu Verlusten führen. [2]

Ein weiteres Problem ist, dass diese Strategien sehr von der Auswahl der Zeiträume für die gleitenden Durchschnitte abhängen. Durch eine schlechte Wahl der Zeiträume kann es dazu kommen, dass die Strategie nicht die Erwartungen erfüllt. [12]

1.4.3 Ausbruchsstrategien

Bei Ausbruchsstrategien wird davon ausgegangen, dass der Aktienkurs aus gesetzten Grenzen ausbricht und sich in die Richtung des Ausbruchs weiterbewegt. Zur Bestimmung der Grenzen werden häufig Hoch- und Tiefpunkte in der Kursentwicklung in einem gewissen Zeitrahmen verwendet. Ausbrüche über die Obergrenze werden als Kaufsignal gewertet, während Ausbrüche unter die Untergrenze als Verkaufssignal gewertet werden. [17] [1]

Mit einer optimierten Ausbruchsstrategie kann sogar der S&P 500 Index in einem Backtest übertroffen werden. Im Zeitraum vom Jänner 2016 bis Dezember 2023 konnte der S&P 500 198% Rendite erzielen, während eine optimierte Ausbruchsstrategie 1637% Rendite erzielen konnte. Die Strategie war Risikoreicher als der S&P 500, konnte aber durch das höhere Risiko auch eine deutlich höhere Rendite erzielen. [31]

Ein Problem bei Ausbruchsstrategien ist, dass es zu Fehlsignalen kommen kann. Dies passiert, wenn eine der Grenzen kurzfristig durchbrochen wird, ohne weiter in die durchbrochene Richtung zu laufen. Durch diese Fehlsignale kann es dazu kommen das bei Spitzen gekauft und in Tälern verkauft wird, was zu Verlusten führt. [24]

1.5 WASM

WebAssembly (WASM) ist ein binäres Instruktionsformat für das Web. Es wurde dafür entwickelt, Webanwendungen schneller und effizienter ausführen zu können und diese dabei gleichzeitig portable zu halten. WebAssembly kann in ~~den~~ verschiedensten höheren Programmiersprachen wie C, C++, Rust und Go geschrieben werden und wird von allen modernen Webbrowsern unterstützt. Der erstellte Quellcode kann anschließend mit einem Compiler, beispielsweise Emscripten, in Bytecode übersetzt werden und durch Aufrufen mit JavaScript in einem Browser ausgeführt werden. [29]
Je nach Browser gibt es aber Unterschiede, wie schnell der WebAssembly-Code ausgeführt werden kann. Für Desktop-Anwendungen erzielt FireFox die beste Ausführungszeit, während Edge am langsamsten ist. Dafür ist die Ausführung von reinem JavaScript im Durchschnitt in FireFox langsamer als reines WebAssembly. Während der Laufzeit weist WebAssembly aber im Vergleich zu JavaScript einen vielfach höheren Speicherverbrauch auf. In allen Desktop-Browsern zeigt sich, dass WebAssembly deutlich mehr Speicher verbraucht als JavaScript (3,39x auf Chrome, 4,93x auf Firefox und 3,44x auf Edge). [30]
Da WebAssembly nicht direkt mit dem Browser interagiert, gibt es einige Einschränkungen. So hat WebAssembly beispielsweise keinen direkten Zugriff auf das Document Object Model (DOM) eines Webbrowsers. Ebenfalls gibt es keinen automatischen Garbage Collector, was zu Folge hat, dass der Entwickler sich selbst um die Speicherverwaltung kümmern muss. [29]
Desweitern können Funktionen in WebAssembly nur einzelne Werte und keine Verbunde oder Objekte zurückgeben. Damit ist also weniger Umfang als in JavaScript gegeben. Es kann desweitern auch nicht auf Umgebungsvariablen zugegriffen werden. [15] [16]

Anhang A

Technische Informationen

Anhang B

Ergänzende Inhalte

Auflistung der ergänzenden Materialien zu dieser Arbeit, die zur digitalen Archivierung an der Hochschule eingereicht wurden (als ZIP-Datei).

B.1 PDF-Dateien

Pfad: /

thesis.pdf . . . . . Finale Master-/Bachelorarbeit (Gesamtdokument)

B.2 Mediendaten

Pfad: /media

*.ai, *.pdf . . . . . Adobe Illustrator-Dateien
*.jpg, *.png . . . . . Rasterbilder
*.mp3 . . . . . Audio-Dateien
*.mp4 . . . . . Video-Dateien

B.3 Online-Quellen (PDF-Kopien)

Pfad: /online-sources

Reliquienschrein-Wikipedia.pdf [**WikiReliquienschrein2023**]

Anhang C

Fragebogen

Anhang D

LaTeX-Quellcode

Quellenverzeichnis

Literatur

[1] Madhav Agarwal. *Breakout Pattern - Meaning and Trading Strategy*. Accessed: 2025-09-19. 2024. URL: https://www.wrightresearch.in/blog/breakout-pattern-meaning-and-trading-strategy/ (siehe S. 6).

[2] P. Arumugam und R. Saranya. „An Empirical Analysis of Trading Strategy Based on Simple Moving Average Crossovers“. *ICTACT Journal on Management Studies* 3.1 (2017), S. 423–426. DOI: 10.21917/ijms.2017.0056 (siehe S. 6).

[3] David H. Bailey u. a. „Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance“. *Notices of the American Mathematical Society* 61.5 (2014), S. 458–471. URL: https://www.ams.org/notices/201405/rnoti-p458.pdf (siehe S. 1).

[4] David H. Bailey u. a. „The Probability of Backtest Overfitting“. *Journal of Computational Finance (Risk Journals)* (Feb. 2015). Forthcoming. DOI: 10.2139/ssrn.2326253 (siehe S. 1, 2).

[5] Andrew Baronick. *BacktestJS Example Result*. Accessed: 2025-09-29. 2024. URL: https://backtestjs.com/docs/trading-strategy-results/view-trading-strategy-results/ (siehe S. 5).

[6] Andrew Baronick. *BacktestJS Website*. Accessed: 2025-09-29. 2024. URL: https://backtestjs.com/ (siehe S. 4).

[7] John Bollinger. „Using bollinger bands“. *Stocks & Commodities* 10.2 (1992), S. 47–51 (siehe S. 5).

[8] WILLIAM BROCK, JOSEF LAKONISHOK und BLAKE LeBARON. „Simple Technical Trading Rules and the Stochastic Properties of Stock Returns“. *The Journal of Finance* 47.5 (1992), S. 1731–1764. DOI: https://doi.org/10.1111/j.1540-6261.1992.tb04681.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1992.tb04681.x (siehe S. 6).

[9] Stephen J. Brown u. a. „Survivorship Bias in Performance Studies“. *The Review of Financial Studies* 5.4 (Mai 2015), S. 553–580. DOI: 10.1093/rfs/5.4.553. eprint: https://academic.oup.com/rfs/article-pdf/5/4/553/24417324/050553.pdf (siehe S. 2).

[10] Mark M. Carhart u. a. „Mutual Fund Survivorship“. *The Review of Financial Studies* 15.5 (Juni 2015), S. 1439–1463. DOI: 10.1093/rfs/15.5.1439. eprint: https://academic.oup.com/rfs/article-pdf/15/5/1439/24432690/151439.pdf (siehe S. 2).

[11] Gilles Daniel, Didier Sornette und Peter Wohrmann. *Look-Ahead Benchmark Bias in Portfolio Performance Evaluation*. 2008. arXiv: 0810.1922 [q-fin.PM]. URL: https://arxiv.org/abs/0810.1922 (siehe S. 2).

[12] Fernando F. Ferreira, A. Christian Silva und Ju-Yi Yen. „Detailed Study of a Moving Average Trading Rule“. *Quantitative Finance* 18.9 (2018), S. 1599–1617. DOI: 10.1080/14697688.2017.1417621 (siehe S. 6).

[13] Luis A. Gil-Alana. „Mean reversion in the real exchange rates“. *Economics Letters* 69.3 (2000), S. 285–288. DOI: https://doi.org/10.1016/S0165-1765(00)00318-9 (siehe S. 5).

[14] Ikhlaas Gurrib. „The Moving Average Crossover Strategy: Does It Work for the S&P500 Market Index?“ *Global Review of Accounting and Finance* 7.1 (2016), S. 92–107. DOI: 10.2139/ssrn.2578302 (siehe S. 6).

[15] Andreas Haas u. a. „Bringing the web up to speed with WebAssembly“. In: *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2017. Barcelona, Spain: Association for Computing Machinery, 2017, S. 185–200. DOI: 10.1145/3062341.3062363 (siehe S. 7).

[16] Andreas Haas u. a. „Bringing the web up to speed with WebAssembly“. *SIGPLAN Not.* 52.6 (Juni 2017), S. 185–200. DOI: 10.1145/3140587.3062363 (siehe S. 7).

[17] Ulf Holmberg, Carl Lönnbark und Christian Lundström. „Assessing the profitability of intraday opening range breakout strategies“. *Finance Research Letters* 10.1 (2013), S. 27–33. DOI: https://doi.org/10.1016/j.frl.2012.09.001 (siehe S. 6).

[18] Jing-Zhi Huang und Zhijian (James) Huang. „Testing moving average trading strategies on ETFs“. *Journal of Empirical Finance* 57 (2020), S. 16–32. DOI: https://doi.org/10.1016/j.jempfin.2019.10.002 (siehe S. 6).

[19] Quantopian Inc. *Zipline Trader Website*. Accessed: 2025-09-30. 2020. URL: https://zipline-trader.readthedocs.io/en/latest/index.html (siehe S. 4).

[20] Quantopian Inc. und Stefan Jansen. *Zipline Reloaded Website*. Accessed: 2025-09-30. 2020. URL: https://zipline.ml4trading.io/ (siehe S. 4).

[21] Mark Leeds. *Bollinger Bands Thirty Years Later*. 2013. arXiv: 1212.4890 [stat.AP]. URL: https://arxiv.org/abs/1212.4890 (siehe S. 5).

[22] Andrew W. Lo und A. Craig MacKinlay. *Data-snooping Biases in Tests of Financial Asset Pricing Models*. Techn. Ber. No. 3020-89-EFA. latest revision: November 1989. Massachusetts Institute of Technology, Sloan School of Management, 1989. URL: http://hdl.handle.net/1721.1/2249 (siehe S. 2).

[23] Richard Martin und Torsten Schöneborn. *Mean Reversion Pays, but Costs*. 2011. arXiv: 1103.4934 [q-fin.TR]. URL: https://arxiv.org/abs/1103.4934 (siehe S. 5).

[24] Cory Mitchell. *Failed Break: What It Is, How It Works, Example*. Accessed: 2025-09-19. 2023. URL: https://www.investopedia.com/terms/f/failedbreak.asp (siehe S. 6).

[25] James M. Poterba und Lawrence H. Summers. „Mean reversion in stock prices: Evidence and Implications“. *Journal of Financial Economics* 22.1 (1988), S. 27–59. DOI: https://doi.org/10.1016/0304-405X(88)90021-9 (siehe S. 6).

[26] Daniel Rodriguez. *Backtrader Website*. Accessed: 2025-09-29. 2015. URL: https://www.backtrader.com/ (siehe S. 3).

[27] Guanru Su. „Analysis of the Bollinger Band Mean Regression Trading Strategy“. In: *Proceedings of the 3rd International Conference on Economic Development and Business Culture (ICEDBC 2023)*. Atlantis Press, 2023, S. 95–102. DOI: 10.2991/978-94-6463-246-0_11 (siehe S. 5).

[28] Ryan Sullivan, Allan Timmermann und Halbert White. „Data-Snooping, Technical Trading Rule Performance, and the Bootstrap“. *The Journal of Finance* 54.5 (1999), S. 1647–1691. DOI: https://doi.org/10.1111/0022-1082.00163. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0022-1082.00163 (siehe S. 2).

[29] WebAssembly Community Group. *WebAssembly*. https://webassembly.org/. Accessed: 2025-09-07. 2025 (siehe S. 7).

[30] Yutian Yan u. a. „Understanding the performance of webassembly applications“. In: *Proceedings of the 21st ACM Internet Measurement Conference*. IMC '21. Virtual Event: Association for Computing Machinery, 2021, S. 533–549. DOI: 10.1145/3487552.3487827 (siehe S. 7).

[31] Carlo Zarattini, Andrea Barbon und Andrew Aziz. *A Profitable Day Trading Strategy For The U.S. Equity Market*. Research Paper 24-98. Swiss Finance Institute, Feb. 2024. DOI: 10.2139/ssrn.4729284 (siehe S. 6).

Software

[32] Andrew Baronick. *BacktestJS GitHub*. Version 1.0.0. Accessed: 2025-09-29. 2024. URL: https://github.com/andrewbaronick/BacktestJS (siehe S. 4).

[33] Quantopian Inc. *Quantopian PyFolio GitHub*. Version 0.9.2. Accessed: 2025-09-30. 2015. URL: https://github.com/quantopian/pyfolio (siehe S. 3).

[34] Quantopian Inc. *Zipline GitHub*. Version 1.4.1. Accessed: 2025-09-30. 2020. URL: https://github.com/quantopian/zipline (siehe S. 4).

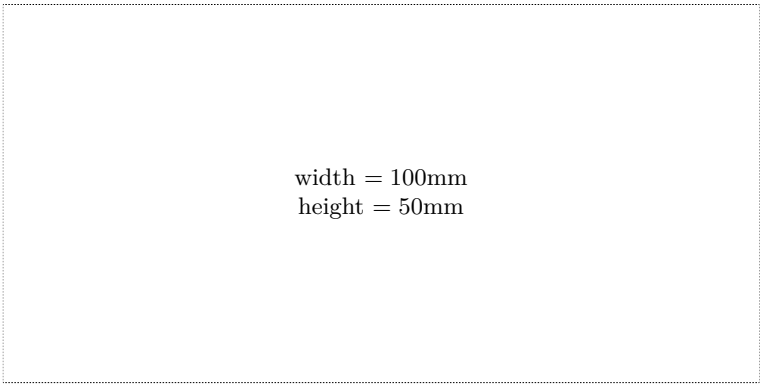
[35] Quantopian Inc. und Stefan Jansen. *Zipline Reloaded GitHub*. Version 3.1.1. Accessed: 2025-09-30. 2020. URL: https://github.com/stefan-jansen/zipline-reloaded (siehe S. 4).

[36] Quantopian Inc. und Shlomi Kushchi. *Zipline GitHub*. Version 1.6.0. Accessed: 2025-09-30. 2020. URL: https://github.com/shlomiku/zipline-trader (siehe S. 4).

[37] Daniel Rodriguez. *Backtrader GitHub*. Version 1.9.78.123. Accessed: 2025-09-29. 2015. URL: https://github.com/mementum/backtrader (siehe S. 3).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —